Subject: [PATCH 1/7 net-2.6.25] [IPV4]: Fix memory leak on error path during FIB initialization.
Posted by den on Fri, 25 Jan 2008 13:51:57 GMT
View Forum Message <> Reply to Message

```
net->ipv4.fib_table_hash is not freed when fib4_rules_init failed. The problem
has been introduced by the following commit.
commit c8050bf6d84785a7edd2e81591e8f833231477e8
Author: Denis V. Lunev <den@openvz.org>
Date:   Thu Jan 10 03:28:24 2008 -0800

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/fib_frontend.c |   10 +++++++++-
 1 files changed, 9 insertions(+), 1 deletions(-)

diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index d282618..d0507f4 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -975,6 +975,7 @@ static struct notifier_block fib_netdev_notifier = {

 static int __net_init ip_fib_net_init(struct net *net)
 {
+	int err;
 	unsigned int i;

 	net->ipv4.fib_table_hash = kzalloc(
@@ -985,7 +986,14 @@ static int __net_init ip_fib_net_init(struct net *net)
 	for (i = 0; i < FIB_TABLE_HASHSZ; i++)
 		INIT_HLIST_HEAD(&net->ipv4.fib_table_hash[i]);

-	return fib4_rules_init(net);
+	err = fib4_rules_init(net);
+	if (err < 0)
+		goto fail;
+	return 0;
+
+fail:
+	kfree(net->ipv4.fib_table_hash);
+	return err;
 }

 static void __net_exit ip_fib_net_exit(struct net *net)
--
1.5.3.rc5
```

Subject: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by den on Fri, 25 Jan 2008 13:51:59 GMT

View Forum Message <> Reply to Message

I could hardly imagine why sombady needs to assign 0.0.0.0 as an interface
address or interface destination address. The kernel will behave in a strage
way in several places if this is possible, as ifa_local != 0 is considered
as initialized/non-initialized state of the ifa.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv4/devinet.c |   12 ++++++++++++
 1 files changed, 12 insertions(+), 0 deletions(-)

diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 9da4c68..e55c85e 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -534,7 +534,13 @@ static struct in_ifaddr *rtm_to_ifaddr(struct nlmsghdr *nlh)
  ifa->ifa_dev = in_dev;

  ifa->ifa_local = nla_get_be32(tb[IFA_LOCAL]);
+ err = -EINVAL;
+ if (ifa->ifa_local == htonl(INADDR_ANY))
+  goto fail_free;
+
  ifa->ifa_address = nla_get_be32(tb[IFA_ADDRESS]);
+ if (ifa->ifa_address == htonl(INADDR_ANY))
+  goto fail_free;

  if (tb[IFA_BROADCAST])
   ifa->ifa_broadcast = nla_get_be32(tb[IFA_BROADCAST]);
@@ -549,6 +555,8 @@ static struct in_ifaddr *rtm_to_ifaddr(struct nlmsghdr *nlh)

  return ifa;

+fail_free:
+ inet_free_ifa(ifa);
 errout:
  return ERR_PTR(err);
 }
@@ -736,6 +744,8 @@ int devinet_ioctl(unsigned int cmd, void __user *arg)
   ret = -EINVAL;
   if (inet_abc_len(sin->sin_addr.s_addr) < 0)
    break;
+  if (sin->sin_addr.s_addr == INADDR_ANY)
+   break;
```

```
   if (!ifa) {
    ret = -ENOBUFS;
@@ -786,6 +796,8 @@ int devinet_ioctl(unsigned int cmd, void __user *arg)
   ret = -EINVAL;
   if (inet_abc_len(sin->sin_addr.s_addr) < 0)
    break;
+  if (sin->sin_addr.s_addr == INADDR_ANY)
+   break;
   ret = 0;
   inet_del_ifa(in_dev, ifap, 0);
   ifa->ifa_address = sin->sin_addr.s_addr;
--
1.5.3.rc5
```

Subject: [PATCH 5/7 net-2.6.25] [IPV4]: fib_sync_down rework.
Posted by den on Fri, 25 Jan 2008 13:52:01 GMT

View Forum Message <> Reply to Message

fib_sync_down can be called with an address and with a device. In reality
it is called either with address OR with a device. The codepath inside is
completely different, so lets separate it into two calls for these two
cases.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/ip_fib.h     |    3 +-
 net/ipv4/fib_frontend.c  |    4 +-
 net/ipv4/fib_semantics.c |  104 +++++++++++++++++++++++----------------------
 3 files changed, 57 insertions(+), 54 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index 9daa60b..1b2f008 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -218,7 +218,8 @@ extern void fib_select_default(struct net *net, const struct flowi *flp,

 /* Exported by fib_semantics.c */
 extern int ip_fib_check_default(__be32 gw, struct net_device *dev);
-extern int fib_sync_down(__be32 local, struct net_device *dev, int force);
+extern int fib_sync_down_dev(struct net_device *dev, int force);
+extern int fib_sync_down_addr(__be32 local);
 extern int fib_sync_up(struct net_device *dev);
 extern __be32  __fib_res_prefsrc(struct fib_result *res);
 extern void fib_select_multipath(const struct flowi *flp, struct fib_result *res);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index d0507f4..d69ffa2 100644
--- a/net/ipv4/fib_frontend.c
```

```
+++ b/net/ipv4/fib_frontend.c
@@ -808,7 +808,7 @@ static void fib_del_ifaddr(struct in_ifaddr *ifa)
       First of all, we scan fib_info list searching
       for stray nexthop entries, then ignite fib_flush.
    */
-   if (fib_sync_down(ifa->ifa_local, NULL, 0))
+   if (fib_sync_down_addr(ifa->ifa_local))
     fib_flush(dev->nd_net);
   }
  }
@@ -898,7 +898,7 @@ static void nl_fib_lookup_exit(struct net *net)

 static void fib_disable_ip(struct net_device *dev, int force)
 {
- if (fib_sync_down(0, dev, force))
+ if (fib_sync_down_dev(dev, force))
   fib_flush(dev->nd_net);
  rt_cache_flush(0);
  arp_ifdown(dev);
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index c791286..5beff2e 100644
--- a/net/ipv4/fib_semantics.c
+++ b/net/ipv4/fib_semantics.c
@@ -1031,70 +1031,72 @@ nla_put_failure:
      referring to it.
    - device went down -> we must shutdown all nexthops going via it.
  */
-
-int fib_sync_down(__be32 local, struct net_device *dev, int force)
+int fib_sync_down_addr(__be32 local)
 {
  int ret = 0;
- int scope = RT_SCOPE_NOWHERE;
-
- if (force)
-  scope = -1;
+ unsigned int hash = fib_laddr_hashfn(local);
+ struct hlist_head *head = &fib_info_laddrhash[hash];
+ struct hlist_node *node;
+ struct fib_info *fi;

- if (local && fib_info_laddrhash) {
-  unsigned int hash = fib_laddr_hashfn(local);
-  struct hlist_head *head = &fib_info_laddrhash[hash];
-  struct hlist_node *node;
-  struct fib_info *fi;
+ if (fib_info_laddrhash == NULL || local == 0)
+  return 0;
```

```
-  hlist_for_each_entry(fi, node, head, fib_lhash) {
-   if (fi->fib_prefsrc == local) {
-    fi->fib_flags |= RTNH_F_DEAD;
-    ret++;
-   }
+ hlist_for_each_entry(fi, node, head, fib_lhash) {
+  if (fi->fib_prefsrc == local) {
+   fi->fib_flags |= RTNH_F_DEAD;
+   ret++;
   }
  }
+ return ret;
+}

- if (dev) {
-  struct fib_info *prev_fi = NULL;
-  unsigned int hash = fib_devindex_hashfn(dev->ifindex);
-  struct hlist_head *head = &fib_info_devhash[hash];
-  struct hlist_node *node;
-  struct fib_nh *nh;
+int fib_sync_down_dev(struct net_device *dev, int force)
+{
+ int ret = 0;
+ int scope = RT_SCOPE_NOWHERE;
+ struct fib_info *prev_fi = NULL;
+ unsigned int hash = fib_devindex_hashfn(dev->ifindex);
+ struct hlist_head *head = &fib_info_devhash[hash];
+ struct hlist_node *node;
+ struct fib_nh *nh;

-  hlist_for_each_entry(nh, node, head, nh_hash) {
-   struct fib_info *fi = nh->nh_parent;
-   int dead;
+ if (force)
+  scope = -1;

-   BUG_ON(!fi->fib_nhs);
-   if (nh->nh_dev != dev || fi == prev_fi)
-    continue;
-   prev_fi = fi;
-   dead = 0;
-   change_nexthops(fi) {
-    if (nh->nh_flags&RTNH_F_DEAD)
-     dead++;
-    else if (nh->nh_dev == dev &&
-      nh->nh_scope != scope) {
-     nh->nh_flags |= RTNH_F_DEAD;
```

```
+ hlist_for_each_entry(nh, node, head, nh_hash) {
+  struct fib_info *fi = nh->nh_parent;
+  int dead;
+
+  BUG_ON(!fi->fib_nhs);
+  if (nh->nh_dev != dev || fi == prev_fi)
+   continue;
+  prev_fi = fi;
+  dead = 0;
+  change_nexthops(fi) {
+   if (nh->nh_flags&RTNH_F_DEAD)
+    dead++;
+   else if (nh->nh_dev == dev &&
+     nh->nh_scope != scope) {
+    nh->nh_flags |= RTNH_F_DEAD;
 #ifdef CONFIG_IP_ROUTE_MULTIPATH
-     spin_lock_bh(&fib_multipath_lock);
-     fi->fib_power -= nh->nh_power;
-     nh->nh_power = 0;
-     spin_unlock_bh(&fib_multipath_lock);
+    spin_lock_bh(&fib_multipath_lock);
+    fi->fib_power -= nh->nh_power;
+    nh->nh_power = 0;
+    spin_unlock_bh(&fib_multipath_lock);
 #endif
-     dead++;
-    }
+    dead++;
+   }
 #ifdef CONFIG_IP_ROUTE_MULTIPATH
-    if (force > 1 && nh->nh_dev == dev) {
-     dead = fi->fib_nhs;
-     break;
-    }
-#endif
-   } endfor_nexthops(fi)
-   if (dead == fi->fib_nhs) {
-    fi->fib_flags |= RTNH_F_DEAD;
-    ret++;
+   if (force > 1 && nh->nh_dev == dev) {
+    dead = fi->fib_nhs;
+    break;
    }
+#endif
+  } endfor_nexthops(fi)
+  if (dead == fi->fib_nhs) {
+   fi->fib_flags |= RTNH_F_DEAD;
+   ret++;
```

```
  }
 }

--
1.5.3.rc5
```

Subject: [PATCH 6/7 net-2.6.25] [NETNS]: Add a namespace mark to fib_info.

Posted by den on Fri, 25 Jan 2008 13:52:02 GMT

View Forum Message <> Reply to Message

This is required to make fib_info lookups namespace aware. In the other case
initial namespace devices are marked as dead in the local routing table
during other namespace stop.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/ip_fib.h      |    1 +
 net/ipv4/fib_semantics.c |    8 ++++----
 2 files changed, 5 insertions(+), 4 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index 1b2f008..cb0df37 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -69,6 +69,7 @@ struct fib_nh {
 struct fib_info {
  struct hlist_node fib_hash;
  struct hlist_node fib_lhash;
+ struct net  *fib_net;
  int   fib_treeref;
  atomic_t  fib_clntref;
  int   fib_dead;
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index 5beff2e..97cc494 100644
--- a/net/ipv4/fib_semantics.c
+++ b/net/ipv4/fib_semantics.c
@@ -687,6 +687,7 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
  struct fib_info *fi = NULL;
  struct fib_info *ofi;
  int nhs = 1;
+ struct net *net = cfg->fc_nlinfo.nl_net;

  /* Fast check to catch the most weird cases */
  if (fib_props[cfg->fc_type].scope > cfg->fc_scope)
@@ -727,6 +728,7 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
   goto failure;
  fib_info_cnt++;
```

```
+ fi->fib_net = net;
  fi->fib_protocol = cfg->fc_protocol;
  fi->fib_flags = cfg->fc_flags;
  fi->fib_priority = cfg->fc_priority;
@@ -798,8 +800,7 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
   if (nhs != 1 || nh->nh_gw)
    goto err_inval;
   nh->nh_scope = RT_SCOPE_NOWHERE;
-  nh->nh_dev = dev_get_by_index(cfg->fc_nlinfo.nl_net,
-          fi->fib_nh->nh_oif);
+  nh->nh_dev = dev_get_by_index(net, fi->fib_nh->nh_oif);
   err = -ENODEV;
   if (nh->nh_dev == NULL)
    goto failure;
@@ -813,8 +814,7 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
  if (fi->fib_prefsrc) {
   if (cfg->fc_type != RTN_LOCAL || !cfg->fc_dst ||
       fi->fib_prefsrc != cfg->fc_dst)
-   if (inet_addr_type(cfg->fc_nlinfo.nl_net,
-       fi->fib_prefsrc) != RTN_LOCAL)
+   if (inet_addr_type(net, fi->fib_prefsrc) != RTN_LOCAL)
     goto err_inval;
  }

--
1.5.3.rc5
```

Subject: [PATCH 7/7 net-2.6.25] [NETNS]: Lookup in FIB semantic hashes taking into account the namespace.
Posted by den on Fri, 25 Jan 2008 13:52:03 GMT

View Forum Message <> Reply to Message

The namespace is not available in the fib_sync_down_addr, add it
as a parameter.

Looking up a device by the pointer to it is OK. Looking up using a result
from fib_trie/fib_hash table lookup is also safe. No need to fix that at all.
So, just fix lookup by address and insertion to the hash table path.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/ip_fib.h      |    2 +-
 net/ipv4/fib_frontend.c  |    2 +-
 net/ipv4/fib_semantics.c |    6 +++++-
 3 files changed, 7 insertions(+), 3 deletions(-)
```

```
diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index cb0df37..90d1175 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -220,7 +220,7 @@ extern void fib_select_default(struct net *net, const struct flowi *flp,
 /* Exported by fib_semantics.c */
 extern int ip_fib_check_default(__be32 gw, struct net_device *dev);
 extern int fib_sync_down_dev(struct net_device *dev, int force);
-extern int fib_sync_down_addr(__be32 local);
+extern int fib_sync_down_addr(struct net *net, __be32 local);
 extern int fib_sync_up(struct net_device *dev);
 extern __be32  __fib_res_prefsrc(struct fib_result *res);
 extern void fib_select_multipath(const struct flowi *flp, struct fib_result *res);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index d69ffa2..86ff271 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -808,7 +808,7 @@ static void fib_del_ifaddr(struct in_ifaddr *ifa)
       First of all, we scan fib_info list searching
       for stray nexthop entries, then ignite fib_flush.
    */
-   if (fib_sync_down_addr(ifa->ifa_local))
+   if (fib_sync_down_addr(dev->nd_net, ifa->ifa_local))
     fib_flush(dev->nd_net);
   }
  }
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index 97cc494..a13c847 100644
--- a/net/ipv4/fib_semantics.c
+++ b/net/ipv4/fib_semantics.c
@@ -229,6 +229,8 @@ static struct fib_info *fib_find_info(const struct fib_info *nfi)
  head = &fib_info_hash[hash];

  hlist_for_each_entry(fi, node, head, fib_hash) {
+  if (fi->fib_net != nfi->fib_net)
+   continue;
   if (fi->fib_nhs != nfi->fib_nhs)
    continue;
   if (nfi->fib_protocol == fi->fib_protocol &&
@@ -1031,7 +1033,7 @@ nla_put_failure:
      referring to it.
    - device went down -> we must shutdown all nexthops going via it.
  */
-int fib_sync_down_addr(__be32 local)
+int fib_sync_down_addr(struct net *net, __be32 local)
 {
  int ret = 0;
  unsigned int hash = fib_laddr_hashfn(local);
```

```
@@ -1043,6 +1045,8 @@ int fib_sync_down_addr(__be32 local)
   return 0;

  hlist_for_each_entry(fi, node, head, fib_lhash) {
+  if (fi->fib_net != net)
+   continue;
   if (fi->fib_prefsrc == local) {
    fi->fib_flags |= RTNH_F_DEAD;
    ret++;
--
1.5.3.rc5
```

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by Daniel Lezcano on Fri, 25 Jan 2008 14:01:34 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> I could hardly imagine why sombady needs to assign 0.0.0.0 as an interface
> address or interface destination address. The kernel will behave in a strage
> way in several places if this is possible, as ifa_local != 0 is considered
> as initialized/non-initialized state of the ifa.

AFAICS, we should be able to set at an interface address to 0.0.0.0, in
order to remove an IP address from an interface and keep this one up.
I see two trivial cases:
 * remove the ipv4 on an interface but continue to use it through ipv6
 * move ipv4 address from the interface to an attached bridge

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by den on Fri, 25 Jan 2008 14:13:32 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Denis V. Lunev wrote:
>> I could hardly imagine why sombady needs to assign 0.0.0.0 as an
>> interface
>> address or interface destination address. The kernel will behave in a
>> strage
>> way in several places if this is possible, as ifa_local != 0 is
>> considered
>> as initialized/non-initialized state of the ifa.
>
> AFAICS, we should be able to set at an interface address to 0.0.0.0, in

> order to remove an IP address from an interface and keep this one up.
> I see two trivial cases:
> * remove the ipv4 on an interface but continue to use it through ipv6
> * move ipv4 address from the interface to an attached bridge

For this case there is an IOCTL/netlink "remove IP address".

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by Daniel Lezcano on Fri, 25 Jan 2008 14:37:53 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> I could hardly imagine why sombady needs to assign 0.0.0.0 as an
>>> interface
>>> address or interface destination address. The kernel will behave in a
>>> strage
>>> way in several places if this is possible, as ifa_local != 0 is
>>> considered
>>> as initialized/non-initialized state of the ifa.
>> AFAICS, we should be able to set at an interface address to 0.0.0.0, in
>> order to remove an IP address from an interface and keep this one up.
>> I see two trivial cases:
>> * remove the ipv4 on an interface but continue to use it through ipv6
>> * move ipv4 address from the interface to an attached bridge
>
> For this case there is an IOCTL/netlink "remove IP address".

That's right. But there are people relying on 0.0.0.0 to remove IP
addresses, especially in the bridge scripts.

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by Daniel Lezcano on Fri, 25 Jan 2008 14:48:05 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> I could hardly imagine why sombady needs to assign 0.0.0.0 as an
>>> interface
>>> address or interface destination address. The kernel will behave in a
>>> strage

>>> way in several places if this is possible, as ifa_local != 0 is
>>> considered
>>> as initialized/non-initialized state of the ifa.
>> AFAICS, we should be able to set at an interface address to 0.0.0.0, in
>> order to remove an IP address from an interface and keep this one up.
>> I see two trivial cases:
>> * remove the ipv4 on an interface but continue to use it through ipv6
>> * move ipv4 address from the interface to an attached bridge
>
> For this case there is an IOCTL/netlink "remove IP address".

And I forgot to mention the general broadcast.
This is need for the dhcp protocol. If you are not able to set your
interface to 0.0.0.0, you will be not able to send a 255.255.255.255
broadcast message to have your IP address.

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by den on Fri, 25 Jan 2008 15:12:41 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Denis V. Lunev wrote:
>> Daniel Lezcano wrote:
>>> Denis V. Lunev wrote:
>>>> I could hardly imagine why sombady needs to assign 0.0.0.0 as an
>>>> interface
>>>> address or interface destination address. The kernel will behave in a
>>>> strage
>>>> way in several places if this is possible, as ifa_local != 0 is
>>>> considered
>>>> as initialized/non-initialized state of the ifa.
>>> AFAICS, we should be able to set at an interface address to 0.0.0.0, in
>>> order to remove an IP address from an interface and keep this one up.
>>> I see two trivial cases:
>>> * remove the ipv4 on an interface but continue to use it through ipv6
>>> * move ipv4 address from the interface to an attached bridge
>>
>> For this case there is an IOCTL/netlink "remove IP address".
>
> And I forgot to mention the general broadcast.
> This is need for the dhcp protocol. If you are not able to set your
> interface to 0.0.0.0, you will be not able to send a 255.255.255.255
> broadcast message to have your IP address.
>

OK. Dave, pls disregard this patch. I suspect that others in the set should not intersect with this one.

To summarize the discussion:
there is the only reason for this assignment: old IOCTL interface does
not have a way to remove IP address except this, though netlink has a
method for it that's why I am a little bit confused :)

This is handled in the __inet_insert_ifa: ifa is just removed there and,
correctly, ifa with 0.0.0.0 address can't exists in the kernel.

Sorry :)

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by Daniel Lezcano on Fri, 25 Jan 2008 15:12:44 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> Denis V. Lunev wrote:
>>>>> I could hardly imagine why sombady needs to assign 0.0.0.0 as an
>>>>> interface
>>>>> address or interface destination address. The kernel will behave in a
>>>>> strage
>>>>> way in several places if this is possible, as ifa_local != 0 is
>>>>> considered
>>>>> as initialized/non-initialized state of the ifa.
>>>> AFAICS, we should be able to set at an interface address to 0.0.0.0, in
>>>> order to remove an IP address from an interface and keep this one up.
>>>> I see two trivial cases:
>>>> * remove the ipv4 on an interface but continue to use it through ipv6
>>>> * move ipv4 address from the interface to an attached bridge
>>> For this case there is an IOCTL/netlink "remove IP address".
>> And I forgot to mention the general broadcast.
>> This is need for the dhcp protocol. If you are not able to set your
>> interface to 0.0.0.0, you will be not able to send a 255.255.255.255
>> broadcast message to have your IP address.
>>
>
> OK. Dave, pls disregard this patch. I suspect that others in the set
> should not intersect with this one.
>
> To summarize the discussion:
> there is the only reason for this assignment: old IOCTL interface does
> not have a way to remove IP address except this, though netlink has a

> method for it that's why I am a little bit confused :)
>
> This is handled in the __inet_insert_ifa: ifa is just removed there and,
> correctly, ifa with 0.0.0.0 address can't exists in the kernel.

Yes, my last statement is false.

Subject: Re: [PATCH 3/7 net-2.6.25] [IPV4]: Prohibit assignment of 0.0.0.0 as interface address.
Posted by Stephen Hemminger on Fri, 25 Jan 2008 16:34:04 GMT
View Forum Message <> Reply to Message

On Fri, 25 Jan 2008 16:51:59 +0300
"Denis V. Lunev" <den@openvz.org> wrote:

> I could hardly imagine why sombady needs to assign 0.0.0.0 as an interface
> address or interface destination address. The kernel will behave in a strage
> way in several places if this is possible, as ifa_local != 0 is considered
> as initialized/non-initialized state of the ifa.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>
>

This is used as a way to bring device up in lots of existing documentation.
So please don't change.

--
Stephen Hemminger <stephen.hemminger@vyatta.com>

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers